

# **Stampen Intelligence**

Kullbäck Christian

Bachelor's thesis in Information Technology

Vaasa 2013

## Table of Contents

1	Introduction.....	1
1.1	Adeprimo – Employer.....	1
1.2	The Application – Sting .....	2
2	Structure and Technology .....	3
2.1	Front End .....	3
2.2	Back End.....	3
2.3	Administrative Interface .....	5
2.4	Problems .....	6
3	Implementation .....	7
3.1	Project Management .....	7
3.1.1	Traditional Project Management.....	7
3.1.2	Agile Project Management.....	8
3.1.3	Methods Used .....	9
3.2	Stages of Development .....	12
3.2.1	Phase 1 .....	12
3.2.2	Phase 2 .....	13
3.2.3	Phase 3 .....	17
3.2.4	Phase 4 .....	19
3.2.5	Phase 5 .....	20
3.3	Sting Rewrite .....	21
4	Results and Discussion .....	24
5	References.....	26

## ABBREVIATIONS

AD	Active Directory
AJAX	Asynchronous Javascript and XML
ASMX	Active Server Method File
CSS	Cascading Style Sheets
DTO	Data Transfer Object
EF	Entity Framework
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IIS	Internet Information Services
JSON	Javascript Object Notation
LINQ	Language Integrated Query
MVC	Model View Controller
POC	Proof of Concept
POF	Proof of concept
REST	Representational state transfer
SASS	Syntactically Awesome Style Sheets
STING	Stampen Intelligence
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WCF	Windows Communication Foundation
WPF	Windows Presentation Foundation
XML	Extensible Markup Language

## BACHELOR'S THESIS

Author: Christian Kullbäck  
Degree Programme: Information Technology  
Supervisors: Kaj Wikman

Title: *Stampen Intelligence*

---

Date 21.05.2013      Number of pages 28      Appendices 2

---

### Abstract

The goal with Sting was to create a web application with the purpose of looking like an iPad application using HTML5. This makes the application platform independent and you are able to use it on any device with an internet connection. The application was developed for Stampen Media Partner and is intended to give a clear and visual overview of the economic situation of the different business areas within the Stampen consortium. An important part of the application is also the ability to upload and share documents with other users. The final result was a web-application that today is up and running in a production environment.

Language: English      Key words: web-application, HTML5, economic, platform-independent

## EXAMENSARBETE

Författare: Christian Kullbäck  
Utbildningsprogram: Informationsteknik  
Handledare: Kaj Wikman

Titel: *Stampen Intelligence*

---

Datum 21.05.2013      Sidantal 28      Bilagor 2

---

### Abstrakt

Målet med Sting var att skapa en webbapplikation vars syfte är att efterlikna en iPad applikation genom att använda sig av HTML5. Detta gör applikationen plattformoberoende och du kan använda den i alla enheter med en internetuppkoppling. Applikationen var utvecklad för Stampen Media Partner för att ge en klar och visuell överblick över den ekonomiska situationen för olika affärsområden inom Stampenkongcernen. En viktig del av applikationen är också möjligheten att kunna skapa och dela dokument med andra användare. Resultatet var en applikation som idag körs inom en produktionsmiljö.

Språk: engelska      Nyckelord: web-applikation, HTML5, ekonomi, plattformsoberoende

# OPINNÄYTETYÖ

Tekijä:

Christian Kullbäck

Koulutusohjelma:

Informaatioteknologia

Ohjaajat:

Kaj Wikman

Nimike: *Stampen Intelligence*

---

Päivämäärä: 21.05.2013

Sivumäärä 28

Liitteet 2

---

## Abstrakti

Stingin tavoitteena oli luoda web-sovellus, joka pyrkii jäljittelemään iPad sovellusta hyödyntämällä HTML5:tä. Tekee sovelluksesta alustariippumaton ja voit käyttää sitä millä tahansa laitteella jossa, on internet yhteys. Sovellus on kehitetty Stampen Media Parterniin tarjoamaan selkeän ja visuaalisen katsauksen taloudellisesta tilanteesta eri liiketoiminta-alueista Stampenin ryhmässä. Tärkeä osa sovelluksessa on myös mahdollisuus luominen ja jakaaminen asiakirjoja muiden käyttäjien kanssa. Tuloksena oli sovellus, joka on parhaillaan käynnissä tuotantoympäristössä.

Kieli: englantia

Avainsanat: web-sovellus, HTML5, talous, alustariippumaton

# **1 Introduction**

## **1.1 Adeprimo – Employer**

Adeprimo AB is the employer who commissioned this thesis project. The company is located in Vaasa, Östersund and Örebro and today it has 60 employees. Adeprimo focuses on digital and mobile products and services. Creating mobile applications, websites and services that will give the customer both market presence and increase in revenue is something that Adeprimo aspires to.

Adeprimo started out in 1994 as a digital agency that delivered websites and creative design. Since then Adeprimo has experienced the IT boom of the 1990s and the crash of 2000. Both of these experiences helped the company to advance and to move into new areas.

In 2006 Adeprimo was bought by the development company mktmedia and started to develop the business platform Tulo commissioned by Stampen, mktmedia and 47 newspapers.

19 years have passed and a lot have changed. Adeprimo strengthened its position as a comprehensive supplier of digital and mobile products by the purchase of the company Leanback with products like E-magin and E-volution.

Since the founding of the company, Adeprimo has gained a lot of market presence but also moved into new exciting areas and today offers several different services such as:

- Digital business concepts & service packaging
- Business- and development projects
- Preliminary studies & prototypes
- Branding & design strategies
- Design assignments
- Management & development of digital solutions

## 1.2 The Application – Sting

The planning and development of Stampen Intelligence (Sting) is the main focus of this thesis. Sting is made to look like an iPad application but is in fact developed in HTML5. This makes the application platform independent and you are able to use it on any device with an internet connection. The application was developed for Stampen Media Partner and is intended to give a clear and visual overview of the economic situation of the Stampen consortium.

The application or service contains three sections. The main part concerns economic data and shows how it is going financially for various business areas and companies within the Stampen Group. It is possible to benchmark different business areas and display useful information, such as who is the director and the turnover per employee.

Besides the viewing of economic data Sting also includes the ability to view and manage collaborations within the consortium. Uploading documents and sharing them with other users is a central part of the application. Users can also comment on the graphical and tabular economic data and the articles within the collaboration section. But all the data within Sting is not visible to everyone since Sting contains a role system. Users may have several roles, and with those roles comes access to certain data and areas belonging to a specific role.



Figure 1. *Stampen Intelligence*



## 2 Structure and Technology

There are a lot ways to design an application. A common way is to separate the different parts of the application into separate sections with the intention of insuring maintainability and reusability. Sting uses this approach instead of the alternative where all of the programming code is put in the same section creating one big mixture of design and functionality logic within the same area. The sections that the application was divided into are listed below.

- Front end (GUI)
- Back end + Database
- Administrative interface

A more detailed description is provided in the sections below.

### 2.1 Front End

The front end works as an interface between the end user and the back end. It functions as a GUI that presents data and information to the user. The front end is also responsible for collecting various inputs from the user that are then conformed into data specifications that the back end can use and store in for example a database.

The Sting front end is almost purely built using JavaScript and HTML5. The methods used for styling the content were CSS/SASS. These methods were chosen because of the rising popularity of JavaScript. Not only because JavaScript is popular when building websites but it is also popular when building mobile applications. Sting was not built using a parser for mobile applications though, but rather built like a website made to look like a mobile application that then uses AJAX to communicate with the back end.

### 2.2 Back End

As mentioned in the above front end section the back end receives information or data from the front end through AJAX requests that are then processed perhaps several times into needed specifications. The back end often consists of some kind of service that receives the information in an independent format. For example the most common formats are XML or JSON. The data is then processed into DTOs that can be stored in a for example a database.

The back end of Sting is a WCF REST Service. WCF was chosen over standard ASMX for its interoperability. With a standard ASMX service you tie yourself to ASP.NET and it is not as loosely coupled from its implementation as WCF. When you implement WCF, you do it against an interface. Then you are able to host the service not just in IIS, but in console applications, WinForms, WPF etc. The downside is that WCF might have a bigger learning curve than ASMX because of the possibilities and complexity that it offers. But all in all WCF is meant to replace ASMX services. Because of that and the above stated facts a WCF service was chosen for the Sting project.

The transfer method used for the web service was REST also known as RESTful, RESTful being a reference to the lightweight nature of the method. When creating a web service there is often a choice between SOAP and REST. Both methods are still widely used and there is no reason why one should not learn to implement them both. The following paragraphs will compare the pros and cons of these two methods.

SOAP has been around for a long time. It is said to be the granddaddy of all web service interfaces. Even though many are steering towards RESTful services there is no need to give up on SOAP just yet. Like many methods it still fills its purpose. It has been around for so long and many large systems are still using it. The problem is that a transfer to a new system would mean a total restructure of existing systems. Compared to its counterpart REST, SOAP also uses more overhead when every call has to be incased in an envelope. This of course can be an advantage when there is a strict standard that you can follow. If there is a need for asynchronous processing and invocation, SOAP can provide both reliability and security. Also SOAP has the ability to use almost any type of transport to perform requests. For example REST uses only HTTP/HTTPS but SOAP can travel over for example SMTP (Simple Mail Transfer Protocol) and even JMS (Java Messaging Service).

REST maybe has not been around for so long but it certainly has gained popularity. Probably because it is quite easy to understand and that it is very approachable since it uses standard HTTP/HTTPS calls to make its requests. That makes REST the perfect option if the application handles stateless CRUD (Create, Read, Update and Delete) operations. If there is a need for stateful operations then perhaps SOAP is the better alternative. REST is also capable of handling several formats, not just XML data that SOAP is bound to. REST can be used with not just XML, but also JSON, HTML and even ordinary text. This makes REST a good alternative if you have a limited amount of bandwidth and resources. And since REST uses URIs to make the requests it is quite easy to develop and test applications since all you need is a browser. Also, when you are dealing with applications that require caching REST is a great option

thanks to the stateless nature of the protocol. Since the Sting application handles CRUD operations and a lot of caching methodology is involved, REST seemed like the best approach to creating the web service.

The data transfer format used in Sting is called JSON (JavaScript Object Notation). It can be compared to XML in the same way as you can compare REST to SOAP. JSON is meant to be a fat-free alternative to XML. While XML is rather efficient in being a document exchange format, it is not as efficient in being a data exchange format. The difference is that XML handles only text where JSON can handle numbers, text and arrays. This makes it perfect for object-oriented programming as it is easy to map JSON to different DTOs. The XML structure is based on elements that can be nested and attributes that cannot. This requires a translation of the data structure to a document structure. That can prove inefficient when dealing with a lot of different data structures. Since Sting is an object-oriented application with a lot of data structures and a front end that uses mostly JavaScript, it seemed only reasonable to use JSON as a data exchange format.

```
{
  lastMonth: "5",
  forecastMonth: "7",
  timestamp: "0201212121224366434",
  - kpi: [
    "AS1450",
    "AS2174",
    "AS1698"
  ],
  startgraph: "AS1450",
  - areas: [
    - {
      id: "ST02",
      name: "ABC Group",
      color: "red",
      - about: {
        boss: "Anders Svensson",
        employees: 3163,
        infotext: "Description Group ABC",
        turnoverperemployee: 135
      },
      - subareas: [
        - {
          id: "ST010",
          name: "CDE Company",
          color: "red",
          - about: {
            boss: "Sven Andersson",
            employees: 1652,
            infotext: "Description CDE Company",
            turnoverperemployee: 162
          },
          - subareas: [
            - {
              id: "ST020",
              name: "FGH Company",
              color: "red",
              - about: {
                boss: "Sven Andersson",
                employees: 359,
                infotext: "Description FGH Company",
                turnoverperemployee: 270
              },
            },
          ],
        },
      ],
    },
  ],
}
```

Figure 2. JSON See Appendix 1 for enhancement

## 2.3 Administrative Interface

An administrative interface is often included in larger applications. This often consists of methods for managing end user accounts or other application-specific content and tasks. For instance in Sting a user with administrative privileges is able to manage users and their roles but is also able to post new articles and collaborations.

For the administrative interface ASP.NET webforms were used because this was a familiar technique within the team. Also because the deadline was closing in and since ASP.NET is a good way of building a small internal application quickly, it felt like the weapon of choice. The administrative side of Sting communicates in the same way with the back end as the front end does by sending and receiving RESTful requests in the form of JSON objects that are then reconstructed into different DTOs depending on the data transmitted.

Alternatives to ASP.NET could have been MVC with a combination of EF. MVC and EF have gained increasing popularity in web development, even though they were originally intended for desktop applications. The structured pattern of MVC makes it a great alternative for scalable applications. MVC works by updating the view, which is the GUI visible to the user. This is done by sending commands to the controller that in turn manipulates the model that most often is connected to a database or service.

## **2.4 Problems**

Problems that the team faced during the development of Sting were perhaps mostly server-related during the start of the project. The long distance communication with the server provider that had to configure all the permissions and install necessary software was perhaps a more tedious task than expected as the team were not sure how the host had set up access to Active Directory and which ip addresses were connected to which server. Also when trying to reach any personel at the hosting company it could take a good long while before a response was received.

Later on in the development it was perhaps mostly the unfamiliarity with the new technologies that put a break on the development. However these kinds of problems always sort themselves out after some self-education trough reference books and a couple of well formulated searches using Google. A couple of books used for reference were Javascript The Definitive Guide by David Flanagan and Visual C# 2010 by John Sharp. The technologies used did not present much of a problem since most of the team was used to C# .NET and a large part of the team also had previous experience using JavaScript. So problems like syntax and other programming language specific problems were not as big an issue as perhaps server-related problems.

## **3 Implementation**

### **3.1 Project Management**

Like any other project, one of the most important aspects is how you manage the project. How you reach estimates and compromises and how you plan the development and implementation so that you reach your goal with all parties feeling good about the result. Often if a project is mismanaged, you wind up with unreasonable timelines, people not knowing their respective tasks and, in the end, unsatisfactory results.

There are many different methodologies and practices to project management. And one can probably not say which one is the best. It all depends on the company and the type of projects that the company deals with. Often it depends on the size of the project, factors within the team and also the experience and competence within the team. It basically boils down to two types of project management.

#### **3.1.1 Traditional Project Management**

In traditional project management supervision plays a large role in the development and planning of a project. Also when doing time estimates with traditional methods you often have a project manager that, with the help of his team, tries to break down a structure or task list for the project. When a list is created then the subject matter experts try to create an estimate for each individual task /3/. This is different from agile management where you try to estimate whole features instead of small tasks. Also, in traditional management when you have reached a time plan you do not deviate from it. Of course the actual time spent and the idealized time will not be the same when unforeseen changes occur, the customer wants to implement features that were not in the original estimates etc. This agile management deals with by being more fluid in development. Changes are easier to implement thanks to the feature-based planning in agile methods. In traditional management, plans and estimates are typically made only once, in a so called “big band” and the rest of the project is spent adjusting to reality. Project managers, rather than working with their team to facilitate more efficient methods of achieving their goals are instead tucked away in an office, probably with a Gantt Chart trying to conform to the reality of development. There are many other methods that are completely alike and different to traditional management, but since agile management is one of the most popular, the upcoming sections will deal with agile methods and their pros and cons.

### **3.1.2 Agile Project Management**

The second most used method is agile project management, which is probably the most popular method in software development. It differs from traditional management in the way that it requires highly motivated individuals from relevant business areas with both supplier and customer input. In agile methods you do not specify the time until delivery in months but in weeks. This is because the projects are submitted in stages instead of one finalized product. Agile management is therefore a variant of iterative management, where the same concept of stages is implemented. This of course puts quite a strain on the people involved since the methods used focus on producing results rather quickly.

In agile development every iteration is an opportunity to reconsider the plan and adjust to reality for the next iteration. Agile methods go by the philosophy that a good plan today is better than a perfect plan tomorrow. This is because there is no such thing as a perfect plan and that it is better to get started with a basic plan that evolves. This also gives the customer the possibility to see some results in the form of a working product or prototype rather quickly instead of getting some half-baked answer that the project is still in the planning stages. And compared to traditional management where everything is planned into the last detail, agile management focuses on feature-based development. And the fact is that most of the time stakeholders and customers do not care about the tasks, nor do they understand them, as much as they care about the final result. This is why stakeholders, customers and sponsors are not brought down to the detail level of a feature, instead they participate in defining the product features as a whole.

Not only are agile projects estimated and developed differently than traditional projects, but they are also planned at different time levels. Many agile practitioners talk of the “planning onion” with planning by the day at the center, expanding out to planning at the iteration, release and project level. The daily meeting, an integral element of agile practice, allows teams to plan in a very granular manner for the results that they plan to deliver that day, and, due to its immediacy, estimates for this work are much more likely to be more accurate.

### 3.1.3 Methods Used

#### 3.1.3.1 Kanban

Agile methods were used in the planning and development of Sting, this because Adeprimo focuses on agile project management and development methods by using a combination of Kanban and Scrum. Kanban is an agile project management method and Scrum is an agile software development method. Kanban focuses on six core practices, which are /4/;5/.

1. Visualize

The workflow of knowledge is inherently visible. Visualizing the workflow and understanding the workflow make it easier to implement changes.

2. Limit WIP

Limiting work-in-process implies that a pull system is implemented on parts or on all parts of the workflow. Work in process refers to the company's finalized yet undelivered products.

3. Manage flow

The flow of work must in each state be monitored, measured and reported. By actively managing the flow the continuous, incremental and evolutionary changes to the system can be evaluated to have positive or negative effects on the system.

4. Make policies explicit

Without an explicit understanding of how things work and how work is actually done, any discussion of problems tends to be emotional, anecdotal and subjective. With an explicit understanding it is possible to move to a more rational, empirical, objective discussion of issues. This is more likely to facilitate consensus around improvement suggestions.

5. Implement feedback loops

Collaboration to review flow of work and demand versus capability measures, metrics and indicators coupled with anecdotal narrative explaining notable events is vital to enabling evolutionary change. Organizations that have not implemented the second level of feedback - the operations review - have generally not seen process improvements beyond a localized team level. As a result, they have not realized the full benefits of Kanban observed elsewhere.

6. Improve collaboratively, evolve experimentally

The Kanban method encourages small continuous, incremental and evolutionary changes that stick. When teams have a shared understanding of theories about work, workflow, process, and risk, they are more likely to be able to build a shared comprehension of a problem and suggest improvement actions which can be agreed by consensus.

### **3.1.3.2 Scrum**

As mentioned above the agile method Scrum was used in the development of Sting. Scrum is an iterative and incremental method that focuses on flexible and holistic product development where a team or a unit works together to reach a common goal as opposed to the traditional sequential approach /7/. Scrum can basically be split into a couple of user roles and components that enforce the Scrum method. The most vital roles are product owner, development team and scrum master and some of the components used are meetings and development periods called sprints.

The product owner represents the stakeholders and is also the voice of the customer. He or she is accountable for ensuring that the team delivers the product. The product owner is also most often responsible for writing user stories in cooperation with the customer. A user story is one or more sentences that describe the application flow and what a user needs to do as part of the product functionality. The product owner then prioritizes the user stories and adds them to the product backlog.

The development team is responsible for delivering a potentially shippable product. The team most often consists of 3-9 people with cross-functional skills who do the actual work and implementation of the user stories. Their job is to analyze, design, develop, test, document etc. The development team is also self-organizing as part of the Scrum philosophy, but also interfaces with project management.

The scrum master is responsible for removing impediments to the ability of the team to deliver the sprint goal/deliverables. The scrum master does not function as a team leader but acts as a buffer between the team and any distracting influences. In contrast to the product owner the scrum master does not engage in people management and customer relations but more in the enforcement of the rules of the Scrum method. The scrum master may also be responsible for managing the meetings that are involved in the Scrum method.

In Scrum sprints are basic units of development. A sprint is a time boxed effort where a development period is restricted to a specific duration. The duration is always fixed in advance for each sprint and is normally between one week and one month. Each sprint is preceded by a planning meeting where the



tasks of the sprint are identified and an estimated commitment for the sprint goal is made. The planning is then followed by a review or a retrospective meeting where the progress is reviewed and lessons for the next sprint are identified.

During each sprint a part or portion of the project is finished. Each part contains features that are put in from the product backlog which is also an ordered list of requirements. Which items go into the sprint is determined during the sprint planning and the product owner also informs the team of which features are of the most importance during this meeting. The team then determines and estimates how many of the features they will be able to implement during the sprint. The features that do not go into the sprint are then planned into the next one.

Meetings are also a basic unit of development in Scrum. This is because communication is a big part of the Scrum method. The communication ensures that all of the team members are up to speed on their respective tasks and also on the other team members' tasks.

A daily scrum is the most basic form of a Scrum meeting where all the members of the team hold a short 10-15 minute meeting about what they have been up to the day before and what their day is going to look like. The intention of this meeting is to keep everyone up to date and to ensure that the project is moving forward. If there are complications that affect the whole team, it is possible to bring these issues up during the meeting as long as the meeting does not exceed 15 minutes. If the issue requires more time, a separate meeting has to be called. There are also weekly scrums and monthly scrums. These meetings are of the same structure as the daily scrums except that the team members might deal with lessons learned during that sprint and that the meetings often run longer than 10-15 minutes.

## **3.2 Stages of Development**

### **3.2.1 Phase 1**

#### **3.2.1.1 Description**

Phase 1 of the project started out mostly as planning. This is where the managerial figures in the project met with the contacts within Stampen AB. Where they came to an agreement on the functionality, design and worked out some basic HTMLdrawings. This part of the project was largely focused on the front end side of the application.

#### **3.2.1.2 Implementation**

Since the Vaasa team were not that much involved in phase 1, this will just be a short introduction to the first development stage of Sting. The implementation of phase 1 largely consisted of developing a proof of concept or a POC that can be described as a basic drawing of design and functionality. No back end was implmented, only static data was used that was hardcoded into the front end. In this way the customer was still able to grasp the look and feel of what was to come later on in the project if they were to move ahead with Sting. The customer was able to click around and still see the flow of the application even if there were not any real data yet. In this way the customer could early on decide on key factors and still change their mind on core functionality without having to worry about costly rebuilds of the project.

### 3.2.2 Phase 2

After phase 1 the team started to focus on the server side solutions for the project. This is where items like login and authentication were considered. Below is a listing of several items that were included in the planning of phase 2.

#### 3.2.2.1 Description

- Back end
  - Agreement on system architecture and back end solutions.
  - Automate the upload of JSON files to back end.
  - Ensure user permissions. Login through Stampen Active Directory.
  - Administrative interface for user and role management.
  - Comments.
- Front end
  - Lock tables when scrolling.
  - Fix table row decimal count for several accounts.
  - Link comments to user accounts.
  - Benchmarking. Fix table headers.
  - Benchmarking layout. What info to present.
  - Embed introduction movie of Stampen Group.
  - Comments.
- Conceptual items
  - Provide suggestions for items above, buttons etc.

#### 3.2.2.2 Implementation

When the project started out the team was split up into back end and front end developers. The staff in Vaasa were mainly in charge of developing the back end and the administrative interface. The staff in Östersund were in charge of developing the front end items. However this did change as the project progressed and people were shifted onto other active projects to optimize time and money.

The team in Vaasa started out by planning the database and the back end solutions. As mentioned a large portion of the application is stored in large JSON files. As information like user permissions and roles, comments, notifications etc of course had to be stored in a database since storing user information directly

on the client side would be both unstable and unsecure, a database structure was planned and implemented using Sql Server 2008.

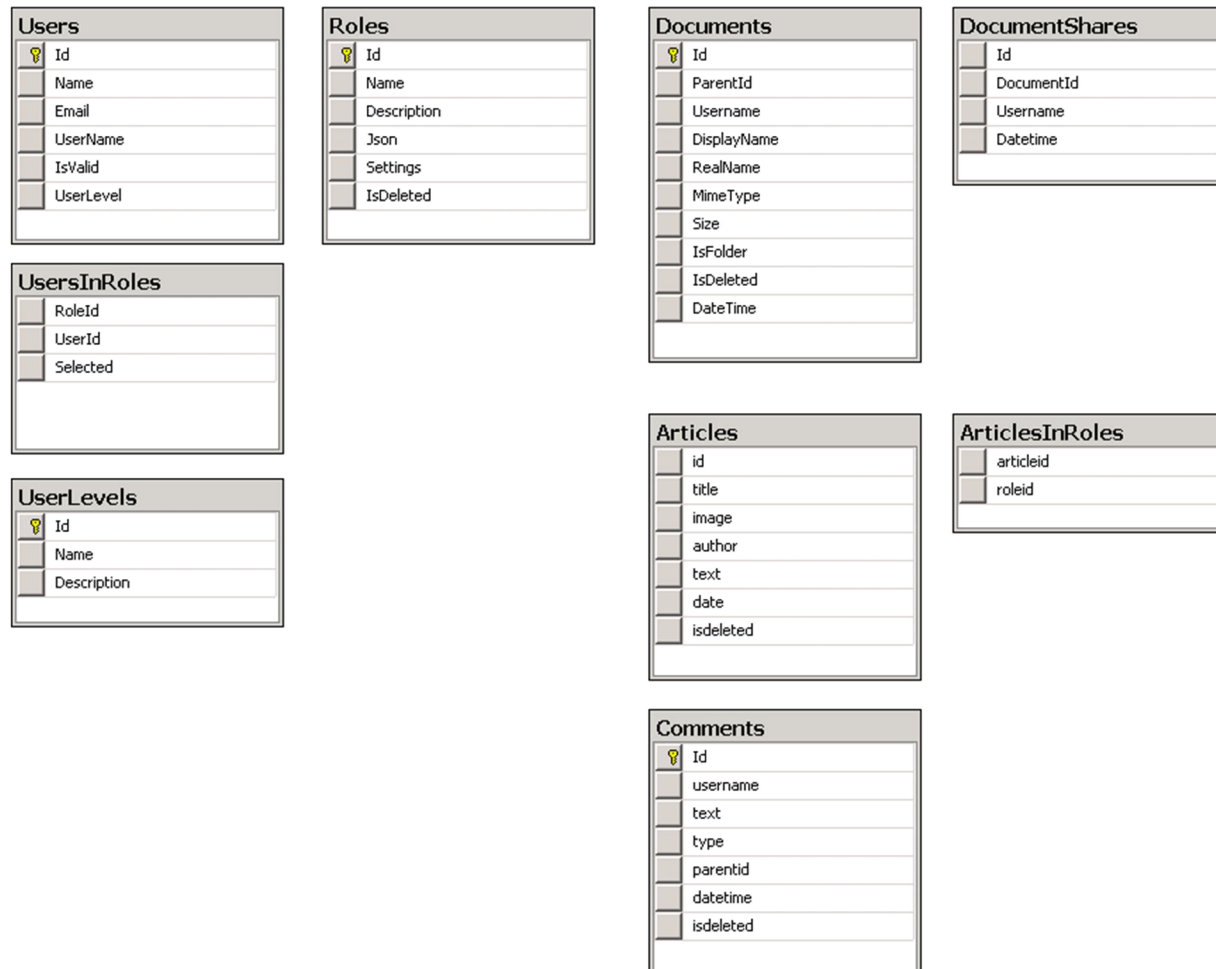


Figure 3. Database Structure

After a database structure was implemented the team then proceeded to research the possibilities for authenticating against The Stampen Group's own network also known as AD (Active Directory). Since Adeprimo is part of Stampen AB and its network, every team member had a login that could be tested within the application. This made the job easier and relieved Stampen server administrators of creating test accounts. A small console application was put together using C# .NET to test and verify the connectivity to Active Directory via the server provided by the server host for Stampen AB. Surprisingly

this worked very well even though none of the team members had developed authentication against AD before using C# .NET and authenticating against a centralized source as AD often brings along some headaches before it is up and running.

The server that was used to host Sting was provided by Stampen AB's server host, Tieto. The server ran on Windows Server 2008 with SQL Server as a database solution. A large part of phase 2 consisted of communication with Tieto's server administrators when for example requesting database accounts and user permissions for the developers in Sting that required constant access to the server. Administrative accounts were set up for the back end developers in the team, but also for some front end developers that needed the possibility of uploading material to the server.

After the authentication against AD had been properly tested the team moved on to creating the WCF REST Service that was going to be used as a conduit between the front end and back end. This was a much more tedious task than expected. As WCF REST uses HTTP requests to communicate it was of course easier to test than for example a web service built using SOAP. Getting the service to respond to the test-requests sent was not the easiest part. A large part of the problems derived from figuring out the structure of the URL (Uniform resource locator) sent to the service. After a couple of hours though, the problem was solved and the team figured out the structure of the request sent to the service. This meant that the team could move on to creating the login requests sent from the application and process them server side.

Sting is as mentioned built around large JSON files that contain the customers' financial data and company information. When a user for instance authenticates against the application a HTTP POST request is sent to the WCF REST Service in the form of JSON-objects. The service then receives the request and processes it according to the URL and the data contained within the request. For instance when a request is sent from the front end through an HTTP request in the form of a POST or GET, the front end first builds a JSON object that represents the data to be sent in text format. The service then receives the request according to the structure and data passes that along to the correct area within the back end that then parses the JSON object according to the object structure of a C# object. The back end then queries the Active Directory with the parsed data containing the username and password supplied from the front end form. Depending on whether the authentication succeeded or not, the back end then queries the database to acquire the appropriate user roles and json data associated with that role. The back end then constructs its own JSON object that is to be sent through the service as a response according to information retrieved from Active Directory to the front end.

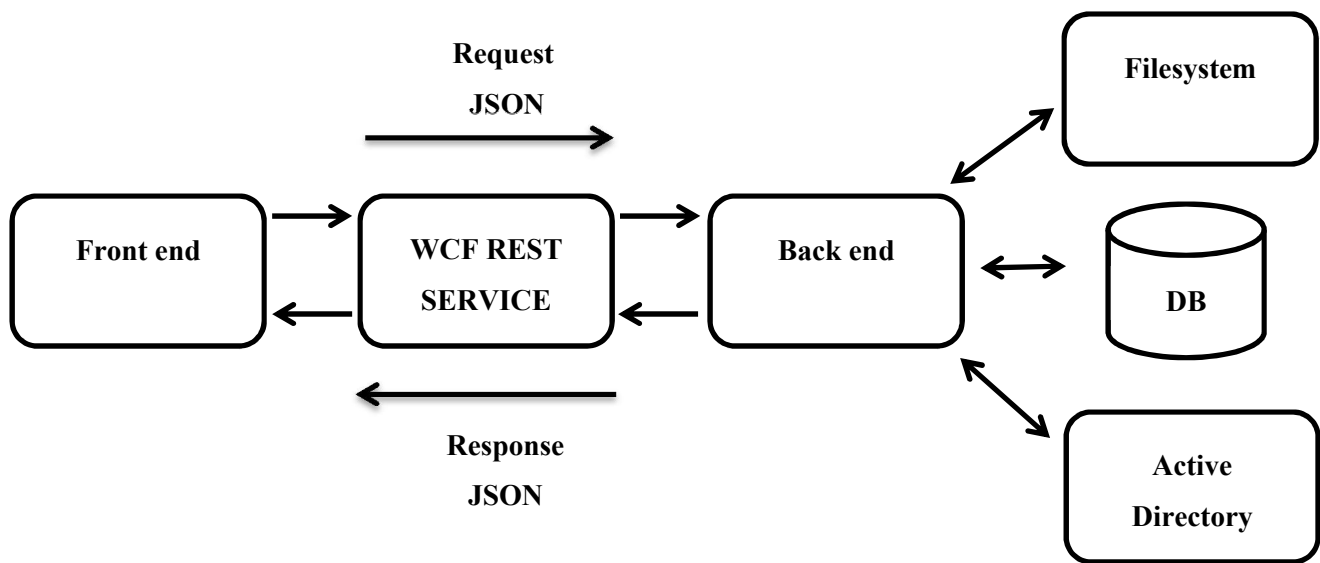


Figure 4. Illustration of communication between the front end and back end

### **3.2.3 Phase 3**

#### **3.2.3.1 Description**

With the completion of phase 2 the team moved on to phase 3 where the following features were requested by the customer:

- Collaborations
  - View collaborations/articles within Sting.
- Documents
  - User is able to create a folder structure of his own.
  - Upload files and share said files.
- Notifications
  - User is able to receive notifications concerning documents and collaborations.
  - View notifications.

#### **3.2.3.2 Implementation**

With the project progressing and all the vital back end building blocks being in place it was getting easier to implement new features through the REST Service. The tasks had moved on from getting the core functionality working to implementing new changes and features that were compatible with the receiving front end. Now basic tasks like communication between the front- and back end seemed insignificant in complexity compared to the new tasks of figuring out and building appropriate DTOs (Data Transfer Objects).

A lot of the work within the back end consisted of creating, building and receiving DTOs that had different tasks depending on the structure and purpose of the request or response. A standard back end feature often consisted of receiving a JSON object from the front end and in turn tearing it down and rebuilding it into a C# compatible object or objects that could then be used to update the database or fetch files from the file system or authenticating against AD.

During the implementation of phase 3 the tasks were quite the same as described in the paragraph above. For instance the collaborations consisted of a request from the front end asking for all collaborations available in the database. This involved building an SQL query against the database that in turn returned a collection of DTO objects that could be parsed into JSON objects ready to be transferred as a response back to the front end.

A user is also able to receive notifications concerning documents that had been shared with that user or when a new article was made available through the collaborations section. This involved sending a timed event request from the front end to the back end with a timestamp that was then checked against the creation date of documents and articles available to the user according to his or her user roles. The back end in turn created the required notification DTO:s to be transferred and then sent them back to the front end, which according to the response notified the user accordingly.

The document section of the application was probably one of the trickier features to implement since all the front end code ran client side and did not have the same access to the server's file system as a language running server side, for example PHP. And since you can't send for example an image that is using AJAX requests it had to be done a bit differently. The image was first converted into a base64 text string that was then parsed into a JSON object that could be transferred and received by the REST Service. Server side, the base64 text string was received and reconstructed to an image that could be saved onto the file system of the server. Also properties like filepath, name, size and user information were then saved onto the database so the file could be used for retrieval when needed.



### **3.2.4 Phase 4**

#### **3.2.4.1 Description**

With phase 4 the project expanded into new features as multiple user roles and article management. Below is a more accurate listing of all the features that were implemented.

- Indicators for the graphical data.
- An article/collaborations manager within the administrative interface of Sting.
  - Create, delete & edit articles from Sting admin as an administrator.
  - Articles bound to user roles.
- Multiple user roles.
  - A user is able to belong to several user roles.

#### **3.2.4.2 Implementation**

In phase 4 a large addition, an article manager was added to the administrative interface. It was implemented so Sting administrators could log on and manage articles/collaborations within Sting. Collaborations in Sting are used as a News feed that describes the ongoing collaborations within the Stampen Group. Users are able to comment on these articles and also receive notifications when another user comments on articles that you have showed interest in and commented on in the past. Articles in Sting are also after phase 4 bound to the same roles as users are. At first all articles were available to all users regardless of role but the customer then realized that perhaps they would like the ability to restrict articles to certain users, not only because of secrecy but perhaps all Sting users would not be interested in the same material as others.

As articles were bound to roles it was obvious that users were going to have to be able to belong to several roles and not be restricted to a single role. This led to the implementation of multiple user roles. When for example an administrator adds a user to Sting, he or she is able to select from a dropdown list, all the roles the user is going to be a member of. The same goes for articles. When an administrator creates an article he or she is able to select the roles associated with that article from a dropdown list.

Indicators were also added to the front end interface, meaning that the tabular economical data was to be dynamically updated according to key indicators that were YTD(Year to date), Periodic and Moving 12 (12 months).

### **3.2.5 Phase 5**

#### **3.2.5.1 Description**

- Documents
  - Sharing with groups and users.
- Administrative Interface
  - Select start page.

#### **3.2.5.2 Implementation**

With phase 5 some new functionality was added to the document section and to the administrative interface. Firstly the customer wanted the ability to share with both users and groups, the groups having the same user roles that were implemented in phase 2 and then extended in phase 4. For this there had to be some changes made to the sharing module on the front end that only contained functionality for sharing with specific users. To the module was added a tab from where a user could select the groups he or she had access to and then share a document with all the users within those groups.

To the administrative interface was added some functionality so that administrators were able to select which page a user would start on according to his or hers set role. Every role contains a JSON file that basically contains all the material found in the application. This material is divided into 5 pages and administrators wanted the possibility to preselect this page from the administrative interface of Sting. This was implemented by adding a radiobutton list to the role editing settings in the administrative interface. These settings were then saved into the database so they could be retrieved at the same time as the front end requests role information from the REST Service and then direct the user to the appropriate page in the application.

## 3.3 Sting Rewrite

### 3.3.1.1 Front End

After the completion of Sting there were discussions about doing a rewrite with the intention of packaging it into a product that could be sold to different customers with the same needs for visual presentation of economic data as the Stampen consortium. This would of course entail more man hours, because of Sting being tailored according to Stampen's Media Partners specific needs.

The rewrite would not only give the team a chance to optimize the existing code but also to consider new development methods and to create a more stable and scalable application that would better suit different businesses with differing economic data. Therefore a study was conducted by the team in order to research and analyze new methods of implementation. All sections of the application would have to be optimized to make the application packagable and customer independent. The section-specific alterations are described below.

For the front end section of the application AngularJS was considered. AngularJS is an open-source JavaScript framework maintained by Google, which assists in running so-called single-page applications. It is made to augment browser-based applications with MVC capability. This is meant to make both testing and development easier and more efficient.

Today Sting is built using mustache tags or logic-less templating. This means that ids are wrapped in curly brackets within the html document and then replaced by data. This of course has its pros and cons, the downside being that a lot of manual coding is required to replace the brackets with data. It is a good and efficient way of templating but in applications that require a lot of dynamic content you quickly notice that you are writing similar pieces of code over and over again. AngularJS addresses this by using two-way data binding. Binding enables the view or the content to be updated whenever the model is altered or changed. This saves both time writing code and avoids error-prone implementations.

### 3.3.1.2 Back End

The back end of Sting is as mentioned a RESTful service, implemented using C# .NET. The techniques used to create the service are quite suitable but some changes could be made to create a more packageable solution. For example the authentication today is done primarily against Active Directory and the service itself is also password protected. The service could possibly be divided into public and private sections where a user would receive a session token when logging in to Sting as this would both increase scalability and security with the option of removing the dependency on Active Directory.

For more efficient database management the application could benefit from the usage of Entity Framework. If you are working with a database first model, which typically is the case, it can be very efficient since the user only has to supply the database structure for EF and it will then accordingly generate a model wrapper for your database that can be used to query your actual database using LINQ. This removes a lot of repetitive code implementation where the user is writing a lot of similar pieces of code over and over again. It also takes care of common mistakes such as spelling errors in your SQL queries and can help to correct common logical errors.

To optimize speed the large JSON files that are used in Sting could perhaps be split into smaller files. For example a single page within the application could be used as a single file. The user could perhaps be presented with the option of loading the whole application or just portions of it. It was also researched whether the loading of files could be done using compression, for instance gzip or zlib. A small test application was developed to see if the browser could handle the processing power required to unzip compressed JSON files. To the team's disappointment it showed that a browser was not capable of efficiently unzipping the files without stalling for a long time and finally not responding at all when presented with larger files than a couple of megabytes.

### **3.3.1.3 Administraive Interface**

The administrative interface of Sting could also benefit from the implementation of similar techniques as in the back end service. However the administrative interface could benefit more from the combination of MVC and EF. Today the admin interface is written using WebForms in C#.NET. WebForms is an efficient way of creating small internal applications such as administrative interfaces but lacks the scalability that MVC implements. Also the new Razor view engine of MVC avoids the usage of server controls that are a large focus in WebForms. These server controls create a lot of hard to understand HTML code and can be hard to debug. Razor focuses on more clean HTML but also uses C# so that the developer can write code within the markup when needed.

The admin interface could also use a tune up of interaction design so its design would be more consistent with the front end that is presented to none-admin users. Today the interface has a look and flow that is quite technically oriented. A standard user that is not used to this could perhaps be confused with this approach.

## 4 Results and Discussion

Sting as a project has been very interesting since it uses a lot of different techniques both client and server side. At least in the beginning of the project there were all the new methods that took some figuring out but as the project progressed it did not take long to become familiar with the structure and implementations of for example other areas that you were not involved in until later in the project. The management of JSON was perhaps one of the more interesting areas as it was quite unfamiliar to me compared to XML. With XML having been the dominant player for so long when talking about data transfer formats it took some time getting used to using JSON. However, after getting over the initial hump of unfamiliarity with using objects in a DTF it quickly grew to being the format of choice.

The way the project was managed using Scrum helped in a lot of ways. For example the constant communication between team members during daily scrums and weekly meetings that summarized the progress and clarified features that the customer had requested helped in keeping the goals clear and much more simpler when the collected thought of the team could provide some insight into features or questions. As this always was a question of interpretation between customer and developer it was also incredibly important to have a conduit for communication between the developers and the customer. Without a product owner keeping the lines of dialog open the project could have been terribly misinterpreted and this could have created a lot of angry stakeholders. Keeping the lines of dialog open within the team also helps the developer in keeping track of his or her own tasks, especially when the team is working from different locations or even different countries as the case was with Sting. Also, everyone knowing their respective tasks helps in optimizing the time spent and does not create unnecessary hours of work that either the company or the customer has to pay for if the project runs over the estimated completion time.

With web development there is the constant problem of keeping yourself up to date with current technologies and developments. So for example when you have finished a large project that took several months to complete there is bound to be new ways to go about implementing the same feature that was implemented in the beginning of the project. Taking these factors into account but also considering scalability and the possibility of selling the same concept to new customers, the idea of rewriting Sting seemed like the logical way to go. And since the Sting project was such an interesting one it would be fun to get a chance to do a rewrite with new optimized methods. The most interesting part will probably be the creation of an application that is so reusable that any customer could use it as long as some minor requirements were met, like server environments and other platform-related details. Not that the current application would be an inferior product and would need tuning but because of new techniques learned along the way that it would be interesting to try out and implement, and if nothing else to see how you have evolved as a developer. If you would do things different the second time, get a better end result thanks to all the experience you have gained from the last time in both design and functionality.

## 5 References

1. AngularJS. Wikipedia. <http://en.wikipedia.org/wiki/AngularJS>. Retrieved 03.03.2013 21:35
2. Flanagan, David. (2011). *Javascript The Definitive Guide* ISBN: 978-0-596-80552-4
3. Freedman, Rick. (2010). Comparing traditional and agile project management estimation techniques.  
<http://www.techrepublic.com/blog/tech-manager/comparing-traditional-and-agile-project-management-estimation-techniques/4357>.  
Retrieved 17.03.2013 16:45
4. Henrik Kniberg. (2007). *Scrum and XP from the Trenches* ISBN: 978-1-4303-2264-1, pp. 4-5
5. Kanban (development). Wikipedia. [http://en.wikipedia.org/wiki/Kanban\\_\(development\)](http://en.wikipedia.org/wiki/Kanban_(development)).  
Retrieved 17.03.2013 19:39
6. Kniberg Henrik & Skarin Mattias. (2010). *Kanban and Scrum making the most of both*.  
ISBN: 978-0-557-13832-6
7. Scrum. Wikipedia. [http://en.wikipedia.org/wiki/Scrum\\_\(development\)](http://en.wikipedia.org/wiki/Scrum_(development)). Retrieved 01.04.2013 18:00
8. Sharp John. (2010). *Visual C# 2010* ISBN: 978-0-7356-2670-6



## Appendix1. JSON test data

```
{
  lastMonth: "8",
  forecastMonth: "7",
  timestamp: "0201212121224366434",
  - kpi: [
    "AS1450",
    "AS2176",
    "AST698"
  ],
  startgraph: "AS1450",
  - areas: [
    - {
      id: "ST02",
      name: "ABC Group",
      color: "red",
      - about: {
        boss: "Anders Svensson",
        employees: 3163,
        infotext: "Description Group ABC",
        turnoverperemployee: 135
      },
      - subareas: [
        - {
          id: "ST010",
          name: "CDE Company",
          color: "red",
          - about: {
            boss: "Sven Andersson",
            employees: 1652,
            infotext: "Description CDE Company",
            turnoverperemployee: 162
          },
          - subareas: [
            - {
              id: "ST020",
              name: "FGH Company",
              color: "red",
              - about: {
                boss: "Sven Andersson",
                employees: 359,
                infotext: "Description FGH Company",
                turnoverperemployee: 270
              },
            },
          ],
        },
      ],
    },
  ],
}
```

## Appendix 2. Sting images



Figure 5. *Sting Login Screen*



Figure 6. *Sting, example of graphical and tabular data*